Matheurística para o problema do safe set

José Paulo de Faria Pedrosa

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DA FACOM-UFMS

Data da Defesa:

Assinatura:_____

Matheurística para o problema do safe set

José Paulo de Faria Pedrosa

Orientador: *Vagner Pedrotti*

Monografia apresentada como requisito parcial para aprovação na Componente Curricular Não-Disciplinar Trabalho de Conclusão de Curso do curso de Graduação em Ciência da Computação, da Universidade Federal de Mato Grosso do Sul.

UFMS - Campo Grande
dezembro/2025

Agradecimentos

*Às mulheres da minha vida — **Florinda, Noêmia, Jane, Daniela e Sofia** — deixo meu mais profundo agradecimento. Obrigado pelo carinho, pelo apoio constante e por acreditarem em mim mesmo quando os desafios pareciam maiores que o caminho.*

*À **Professora Dra. Edna Ayako Hoshino**, registro meus sinceros agradecimentos pelas contribuições substanciais oferecidas ao longo da elaboração deste trabalho. Diversos aprimoramentos metodológicos e escolhas técnicas foram possíveis graças às discussões, análises e sugestões por ela apresentadas. Sua experiência acadêmica, aliada à atenção minuciosa dispensada às revisões e ao desenvolvimento da abordagem adotada, desempenhou papel decisivo para a qualidade final deste TCC. Sou profundamente grato por sua disponibilidade, dedicação e apoio constante.*

*Ao **Professor Dr. Vagner Pedrotti**, meu orientador, registro minha profunda gratidão pela confiança, pela paciência e pela tranquilidade com que sempre conduziu este trabalho. Sua serenidade, sua clareza nas explicações e sua constante disposição em ajudar tornaram toda esta jornada muito mais leve e enriquecedora. Agradeço também por acreditar no meu potencial e por me acompanhar com atenção e generosidade em cada etapa do desenvolvimento deste trabalho.*

Resumo

O problema do *safe set* (SSP) consiste em identificar, em um grafo conexo, um subconjunto de vértices capaz de dominar estruturalmente as componentes adjacentes, garantindo que cada componente interna possua peso maior ou igual ao das componentes externas a ela conectadas. Trata-se de um problema NP-difícil, com aplicações em redes sociais, estabilidade de sistemas vulneráveis e organização de refúgios em edificações. Devido à sua complexidade computacional, métodos exatos apresentam limitações para instâncias de maior porte, o que motiva o uso de estratégias aproximadas. Este trabalho propõe uma matheurística composta por duas etapas: (i) uma fase construtiva baseada em uma heurística gulosa aleatorizada, responsável por gerar soluções iniciais viáveis, e (ii) uma fase de otimização utilizando a abordagem *Large Neighbourhood Search* (LNS), que destrói e reconstrói parcialmente a solução valendo-se do modelo compacto de Programação Linear Inteira proposto por Hosteins (2020). Foram conduzidos experimentos em 120 instâncias pseudoaleatórias da literatura, variando-se o tamanho do grafo, a densidade e as estratégias de seleção de vértices. Os resultados demonstram que a matheurística produz soluções de boa qualidade em tempo reduzido, com redução média de aproximadamente 50% em relação ao valor das soluções iniciais, e desempenho superior em instâncias mais densas e de maior escala. As estratégias com maior número de iterações construtivas (B-10 e F-10) destacaram-se, especialmente a configuração F-10, que obteve as melhores soluções na maioria dos casos. Os resultados confirmam a eficácia da combinação entre heurísticas gulosas e LNS guiado por PLI como alternativa promissora para a resolução do SSP.

Sumário

CAPÍTULO

1

Introdução

1.1 Motivação

O Problema do *Safe Set* (SSP) é um problema de particionamento de grafos que modela situações nas quais se deseja identificar subconjuntos de vértices para ganhar controle de toda a rede. Considerando um grafo simples, não-orientado e conexo, em que os vértices possuem pesos, dizemos que um subconjunto S dos vértices é um *safe set* se, para toda componente conexa C, induzida por S, e toda componente conexa D induzida por $(V \setminus S)$, no qual existe uma aresta entre C e D, a soma dos pesos dos vértices em D não ultrapassa a soma dos pesos dos vértices em C. O objetivo do SSP é determinar um *safe set* de peso mínimo. Na literatura encontramos diversas aplicações práticas para esse problema, por exemplo, identificar comunidades influentes em redes sociais ou identificar grupos majoritários que podem causar instabilidade em redes vulneráveis [Bapat et al. (2016)]. Outra aplicação envolve a distribuição de refúgios temporários em grandes edifícios comerciais [Fujita et al. (2016)]. Nesta aplicação, cada sala segura deve ser acessível a quem está em salas adjacentes, o que implica que sua capacidade deva ser pelo menos igual à capacidade das salas conectadas a ela.

1.2 Objetivos

Este estudo tem como objetivo empregar métodos de metaheurísticas fundamentadas em programação matemática, denominadas matheurísticas, com a finalidade de desenvolver soluções viáveis para o SSP, dentro de um tempo

computacional aceitável (heurísticas). Esse objetivo se revela significativo, uma vez que o SSP é classificado como NP-difícil [Bapat et al. (2016)].

1.3 Trabalhos relacionados

O SSP foi primeiramente apresentado por [Bapat et al. (2016)], sendo, posteriormente, divulgada sua versão com pesos unitários nos vértices por [Fujita et al. (2016)]. Em uma perspectiva computacional, Macambira et al. (2019) sugeriram uma formulação linear inteira, a qual foi solucionada por meio de um algoritmo exato do tipo *Branch-and-Cut*, com apoio de uma heurística gulosa. Já Hosteins (2020) desenvolveu um modelo compacto para o problema, o qual possui um número polinomial de variáveis e restrições, fundamentado em fluxo de rede. Recentemente, um novo algoritmo *Branch-and-Cut* foi apresentado por Malaguti e Pedrotti (2023).

1.4 Organização do texto

O Capítulo 1 expõe a motivação e os objetivos desta pesquisa, bem como os estudos correlatos. O Capítulo 2 estabelece formalmente o problema, apresentando a notação utilizada e a finalidade do estudo. O Capítulo 3 expõe a metodologia empregada e estabelece conceitos significativos para a elaboração do trabalho. O Capítulo 4 expõe a atividade realizada e o modelo de Programação Linear Inteira empregado. O Capítulo 5 expõe o contexto computacional utilizado e os resultados alcançados. Finalmente, o Capítulo 6 apresenta as conclusões e as reflexões finais do trabalho.

CAPÍTULO

2

Descrição do Problema

Uma descrição formal do SSP é apresentada neste capítulo.

2.1 Descrição formal do SSP

Considere $G = (V, E)$ um grafo simples, não orientado e conexo. Um grafo é dito conexo se, para qualquer par $\{z, w\}$ de seus vértices, existe um caminho com extremos z e w. Já o subgrafo induzido por um subconjunto $S \subseteq V$ é aquele obtido a partir de G pela exclusão dos vértices em $V \setminus S$ e de todas as arestas incidentes a eles. As componentes conexas de um grafo são os seus subgrafos conexos maximais. Denotamos por $G[S]$ o subgrafo de G induzido por um subconjunto $S \subseteq V$, e chamamos de P_v a componente conexa de $G[S]$ que contém o vértice $v \in V$, se $v \in S$; ou a componente conexa de $G[V \setminus S]$ que contém v, se $v \notin S$. Diz-se que um subconjunto S é um *safe set* de G se, para toda aresta uv de G, tal que $u \in S$ e $v \notin S$, a condição $|P_u| \geq |P_v|$ é satisfeita. A Figura 2.1 ilustra um exemplo de *safe set*, com indicação de suas componentes.

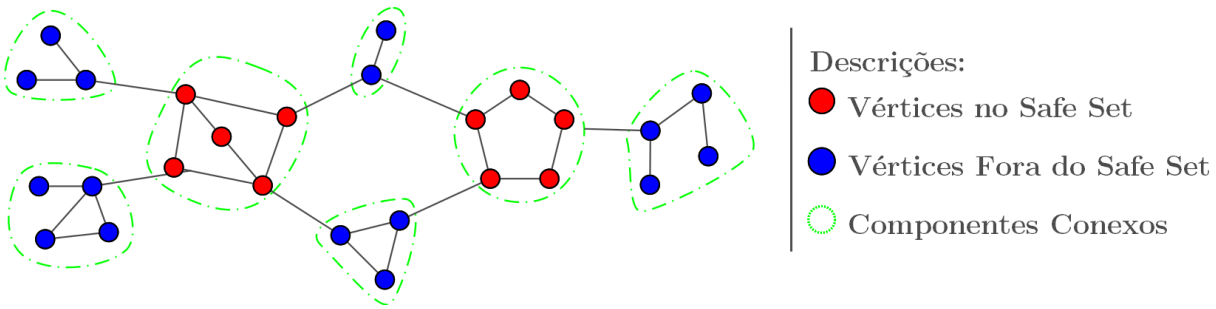

Figura 2.1: Exemplo de *safe set* viável com pesos unitários.

2.2 Descrição formal do Problema do Safe Set com pesos

Para definir o *safe set* com pesos, utilizamos a função $w : V \to \mathbb{R}^+$, que atribui um peso positivo a cada vértice $i \in V$. Para qualquer subconjunto S de V, definimos $w(S)$ como a soma dos pesos de todos os elementos de S. Assim, $w(S) = \sum_{i \in S} w(i)$. Assim, um subconjunto não vazio $S \subseteq V$ é um *safe set* com pesos se, para cada componente conexa C de $G[S]$ e para cada componente conexa D de $G[V \setminus S]$ que é adjacente a C, é satisfeita a desigualdade $w(C) \geq w(D)$. O Problema do *Safe Set* com e sem pesos consiste, respectivamente, em determinar um *safe set* de cardinalidade mínima e um *safe set* de peso mínimo em um grafo.

2.3 O modelo exato de Hosteins para o SSP

O modelo de PLI utilizado neste projeto foi proposto por Hosteins (2020) e fornecido como base para a formulação do problema. A seguir, apresenta-se a formulação matemática do problema.

$$\min \quad \sum_{i \in V} w_i x_i \tag{1}$$

$$\text{s.a.} \quad y_{ij} \geq x_i + x_j - 1 \qquad \{i,j\} \in E \tag{2}$$

$$y'_{ij} \geq 1 - x_i - x_j \qquad \{i,j\} \in E \tag{3}$$

$$y_{ij} \leq x_i \qquad \{i,j\} \in E \tag{4}$$

$$y_{ij} \leq x_j \qquad \{i,j\} \in E \tag{5}$$

$$y'_{ij} \leq 1 - x_i \qquad \{i,j\} \in E \tag{6}$$

$$y'_{ij} \leq 1 - x_j \qquad \{i,j\} \in E \tag{7}$$

$$\sum_{j \in N_i} f_{ij} - \sum_{j \in N_i \cup \{0\}} f_{ji} = -1 \qquad i \in V \tag{8}$$

$$\sum_{i \in V} f_{0i} = n \tag{9}$$

$$f_{0i} \leq (n-1) \sum_{c=1}^{n-1} (t_{ic} + t'_{ic}) \qquad i \in V \tag{10}$$

$$f_{ij} \leq (n-1)(y_{ij} + y'_{ij}) \qquad (i,j) \in E' \tag{11}$$

$$\sum_{c=1}^{n-1} a_{ic} = x_i \qquad i \in V \tag{12}$$

$$\sum_{c=1}^{n-1} a'_{ic} = 1 - x_i \qquad i \in V \tag{13}$$

$$a_{ic} \geq a_{jc} + y_{ij} - 1 \qquad \{i,j\} \in E, c = 1,...,n-1 \tag{14}$$

$$a_{jc} \geq a_{ic} + y_{ij} - 1 \qquad \{i,j\} \in E, c = 1,...,n-1 \tag{15}$$

$$a'_{ic} \geq a'_{jc} + y'_{ij} - 1 \qquad \{i,j\} \in E, c = 1,...,n-1 \tag{16}$$

$$a'_{jc} \geq a'_{ic} + y'_{ij} - 1 \qquad \{i,j\} \in E, c = 1,...,n-1 \tag{17}$$

$$t_{ic} \leq a_{ic} \qquad i \in V, c = 1,...,n-1 \tag{18}$$

$$t'_{ic} \leq a'_{ic} \qquad i \in V, c = 1,...,n-1 \tag{19}$$

$$\sum_{i \in V} t_{ic} \leq 1 \qquad c = 1,...,n-1 \tag{20}$$

$$\sum_{i \in V} t'_{ic} \leq 1 \qquad c = 1,...,n-1 \tag{21}$$

$$v_c = \sum_{i \in V} w_i a_{ic} \qquad c = 1,...,n-1 \tag{22}$$

$$v'_c = \sum_{i \in V} w_i a'_{ic} \qquad c = 1,...,n-1 \tag{23}$$

$$\omega_i \geq v_c - W(1 - a_{ic}) \qquad i \in V, c = 1,...,n-1 \tag{24}$$

$$\omega_i \leq v_c + W(1 - a_{ic}) \qquad i \in V, c = 1,...,n-1 \tag{25}$$

$$\omega_i \leq W x_i \qquad v \in V \tag{26}$$

$$\omega'_i \geq v'_c - W(1 - a'_{ic}) \qquad i \in V, c = 1,...,n-1 \tag{27}$$

$$\omega'_i \leq v'_c + W(1 - a'_{ic}) \qquad i \in V, c = 1,...,n-1 \tag{28}$$

$$\omega'_i \leq W(1 - x_i) \qquad i \in V \tag{29}$$

$$\omega_i \geq \omega'_j - W y'_{ij} \qquad (i,j) \in E' \tag{30}$$

$$\sum_{i \in V} x_i \geq 1 \tag{31}$$

$$y_{ij}, y'_{ij} \in \{0,1\} : \{i,j\} \in E \tag{32}$$

$$a_{ic}, a'_{ic} \in \{0,1\} : i \in V, c = 1,...,n-1 \tag{33}$$

$$\omega_i, \omega'_i \geq 0 : i \in V \tag{34}$$

$$v_c, v'_c \geq 0 : c = 1,...,n-1 \tag{35}$$

$$f_{ij} \in [0, n-1] : i \in V \cup \{0\}, j \in V \tag{36}$$

$$x_i \in \{0,1\} : i \in V \tag{37}$$

$$t_{ic}, t'_{ic} \in \{0,1\} : i \in V, c = 1,...,n-1 \tag{38}$$

Descrição das Variáveis

- x_i: variável binária que assume valor 1 se $i \in S$ (*safe set*) e 0, caso contrário.
- y_{ij}: variável binária que assume valor 1 se $\{i,j\} \in E$ com $i,j \in S$, e 0, caso

contrário.

- y'_{ij}: variável binária que assume valor 1 se $\{i,j\} \in E$ com $i,j \in V \setminus S$, e 0, caso contrário.
- a_{ic}: variável binária que assume valor 1 se o nó $i \in S$ pertence ao componente $c \in \{1,\ldots,n-1\}$.
- a'_{ic}: variável binária que assume valor 1 se o nó $i \in V \setminus S$ pertence ao componente $c \in \{1,\ldots,n-1\}$.
- t_{ic}: variável binária que assume valor 1 se o nó $i \in S$ é o único nó em $c \in \{1,\ldots,n-1\}$ que recebe fluxo não nulo proveniente do nó 0.
- t'_{ic}: variável binária que assume valor 1 se o nó $i \in V \setminus S$ é o único nó em $c \in \{1,\ldots,n-1\}$ que recebe fluxo não nulo proveniente do nó 0.
- f_{ij}: variável contínua de fluxo de $i \in V \cup \{0\}$ para $j \in V$, definida se $i = 0$ ou $(i,j) \in E'$.
- ω_i: peso total do componente em $G[S]$ ao qual o nó i pertence.
- ω'_i: peso total do componente em $G[V \setminus S]$ ao qual o nó i pertence.
- ν_c: peso total do componente $c \in \{1,\ldots,n-1\}$.
- ν'_c: peso total do componente $c \in \{1,\ldots,n-1\}$.

Grupos de Restrições

- Restrições (2)–(7): garantem a consistência das variáveis y e y' em relação às variáveis de decisão x, correspondendo a uma linearização das expressões $y_{ij} = x_i x_j$ e $y'_{ij} = (1-x_i)(1-x_j)$ para $\{i,j\} \in E$.
- Restrição (8): assegura que uma unidade de fluxo proveniente do nó fictício 0 seja absorvida por cada nó do grafo.
- Restrição (9): garante que exatamente n unidades de fluxo sejam fornecidas pelo nó fictício 0, correspondendo ao número total de nós em V.
- Restrições (10): asseguram que o fluxo só pode sair do nó 0 em direção a outro nó se a variável t_{ic} ou t'_{ic} correspondente for não nula.
- Restrição (11): impõe que o fluxo só pode circular em arestas internas a componentes de $G[S]$ ou $G[V \setminus S]$.
- Restrições (12)–(13): forçam que cada nó $i \in V$ seja atribuído a exatamente um componente, seja em $G[S]$ ou em $G[V \setminus S]$.

- Restrições (14)–(17): asseguram a consistência das variáveis a e a' de forma que dois nós pertencentes a um mesmo componente recebam o mesmo rótulo c.

- Restrições (18)–(21): definem a consistência entre variáveis t e a, impondo que em cada componente apenas um nó possa assumir valor não nulo para t, representando o ponto de entrada do fluxo proveniente de 0.

- Restrições (22)–(23): definem o peso total de cada componente $c \in \{1, \ldots, n-1\}$ em função das variáveis a e a'.

- Restrições (24)–(29): vinculam as variáveis ω_i e ω'_i aos pesos ν_c e ν'_c dos componentes, garantindo consistência na atribuição dos pesos aos nós.

- Restrição (30): relaciona as variáveis de peso ω e ω' por meio das variáveis y', garantindo consistência entre os pesos dos componentes em $G[S]$ e $G[V \setminus S]$.

- Restrição (31): garante que pelo menos um nó seja selecionado para compor o *safe set*.

- Restrições (32)–(38): definem os domínios das variáveis binárias e contínuas do modelo (condições de integralidade e não negatividade).

CAPÍTULO

3

Metodologia

Serão apresentados neste capítulo os conceitos fundamentais para a realização deste projeto e a metodologia que foi utilizada.

3.1 Programação linear

A Programação Linear (PL) é uma técnica de otimização amplamente utilizada para resolver problemas de otimização combinatória e alocação de recursos. Tem como objetivo a maximização ou a minimização de uma função linear denominada função objetivo, sujeita a um conjunto de restrições lineares. A programação linear fundamenta-se em princípios matemáticos robustos e é sustentada por algoritmos eficazes, como o método Simplex e as abordagens de pontos interiores, os quais viabilizam a resolução de instâncias de problemas em larga escala.

3.2 Programação linear inteira

Em muitos problemas práticos de otimização, como no caso do SSP, as escolhas a serem realizadas apresentam um caráter binário e dizem respeito, por exemplo, a estabelecer se um vértice deve, ou não, fazer parte de um subconjunto. Em cenários como este, em que as variáveis necessitam ser limitadas a números inteiros, recorre-se a um ramo da PL, denominado Programação Linear Inteira (PLI).

A distinção fundamental entre esses dois modelos está no domínio das variáveis de decisão: enquanto na PL as variáveis podem assumir qualquer valor

dentro do conjunto dos números reais, na PLI estabelece-se a limitação de que essas variáveis devem pertencer ao conjunto dos números inteiros. Essa particularidade torna os modelos de PLI mais apropriados para a representação de problemas nas quais as decisões envolvem quantidades discretas, como alocações, seleções ou combinações de elementos, refletindo de forma mais precisa a essência do problema em estudo.

3.3 Branch-and-Bound

O método *Branch-and-Bound* (B&B) é uma estratégia de busca sistemática utilizada para a resolução de problemas de otimização combinatória. Seu funcionamento baseia-se na divisão do espaço de soluções em subespaços menores (*branching*), e na avaliação de limitantes que permitem eliminar regiões do espaço que não podem conter a solução ótima (*bounding*).

A divisão do espaço de soluções pode ser visualizada como uma árvore de enumeração, na qual o nó raiz denota o problema original, contendo todo o espaço de soluções viáveis, e os nós filhos representam subproblemas, contendo subespaços menores da região de soluções viáveis. Em cada nó, calcula-se um limitante dual — inferior em problemas de minimização ou superior em problemas de maximização — com o objetivo de estimar o melhor valor possível dentro daquele subespaço. Esse limitante pode ser obtido de diferentes maneiras, como pela resolução de uma relaxação linear ou por outras técnicas de relaxação. Um limitante dual global é mantido e ele representa o melhor dos limitantes duais - menor dentre todos os limitantes duais em problema de minimização ou o maior dentre todos os limitantes duais em problemas de maximização. Um limitante primal global também é mantido e refere-se ao valor da melhor solução viável encontrada. Soluções viáveis são encontradas por heurísticas que são executadas em cada nó da árvore ou quando a solução da relaxação linear é também uma solução inteira - em que o valor das variáveis pertencentes ao domínio do conjunto dos inteiros tem um valor inteiro.

Se a solução obtida em um nó é viável e melhora o melhor valor conhecido, ela atualiza o limitante primal global. Caso contrário, o nó pode ser podado por inviabilidade, otimalidade ou por apresentar um limitante dual pior que o limitante primal global, significando que a subárvore enraizada naquele nó não precisa ser explorada porque não produzirá solução com valor melhor do que o já conhecido. Os nós que não foram podados são chamados de nós ativos. A cada iteração, o algoritmo *branch-and-bound* seleciona um dos nós ativos para ser explorado e ramificado, originando novos subproblemas que subdividem o subespaço viável daquele nó.

Esse processo se repete até que todos os nós ativos tenham sido explora-

dos ou podados, assegurando que a melhor solução encontrada seja ótima para o problema original. O desempenho do método depende diretamente da qualidade dos limitantes e das estratégias de seleção dos nós a explorar.

3.4 Matheurísticas

Podemos descrever matheurísticas como métodos de resolução que combinam programação matemática com abordagens heurísticas ou metaheurísticas. Essa abordagem une a precisão dos modelos matemáticos com a eficiência das heurísticas, que facilitam a busca por soluções de forma mais rápida e prática. Esse tipo de método é especialmente utilizado para tratar problemas complexos ou de grande porte, em que a obtenção de soluções exatas acaba sendo inviável devido ao alto custo computacional envolvido.

3.4.1 Greedy Randomized Adaptive Search Procedure (GRASP)

O *Greedy Randomized Adaptive Search Procedure* (GRASP) é uma metaheurística originalmente proposta em Feo e Resende (1989) e posteriormente detalhada em Feo e Resende (1995), que combina estratégias gulosas e aleatórias para gerar soluções. O GRASP é um método iterativo, sendo que cada iteração possui duas fases principais, a **fase de construção** e a fase de **busca local**. Ao fim de cada iteração é gerada uma solução, e ao fim da execução do algoritmo a melhor solução encontrada é retornada.

Pseudocódigo do GRASP

A seguir, apresenta-se o pseudocódigo geral do GRASP:

Algoritmo 1: GRASP

Input: Número de iterações n_{iter}, parâmetro de aleatoriedade α
Output: Melhor solução encontrada

```
1 melhor_solucao ← ∅;
2 for i ← 1 to n_iter do
3 |   solucao ← Construção(α);
4 |   solucao ← Busca_Local(solucao);
5 |   if Custo(solucao) < Custo(melhor_solucao) then
6 |   |   melhor_solucao ← solucao;
7 |   end
8 end
9 return melhor_solucao;
```

3.4.2 Large Neighbourhood Search (LNS)

Large Neighbourhood Search (LNS) é uma matheurística introduzida por Shaw (1998) no contexto de problemas de roteamento de veículos. O objetivo principal é melhorar as soluções iniciais por meio de ciclos repetidos de destruição e reconstrução. É composta por quatro etapas principais:

1. **Destruição**: elimina-se uma parte da solução atual, criando espaço para novas opções. A remoção pode ser realizada de forma aleatória ou orientada por heurísticas.

2. **Reconstrução**: a instância resultante é resolvida novamente, geralmente com o auxílio de um solver ou de heurísticas construtivas, gerando uma nova solução que pode ser melhor que a anterior.

3. **Aceitação**: se a nova solução representa uma melhoria em relação à anterior, ela passa a substituir a solução existente. Em algumas circunstâncias, soluções menos eficientes podem ser toleradas para promover a diversificação da pesquisa.

4. **Iteração**: o processo é realizado repetidamente até que um critério de conclusão seja alcançado (número de iterações, prazo final ou convergência da solução).

Pseudocódigo do LNS

A seguir, apresenta-se o pseudocódigo genérico do LNS:

Algoritmo 2: LNS (Large Neighbourhood Search)

Input: Instância I, número de iterações n_{iter}, parâmetro de aleatoriedade α

Output: Melhor solução encontrada

```
1 melhor_solucao ← ConstroiSolucaoInicial(I);
2 for i ← 1 to n_iter do
3 |   solucao_parcial ← DestroiParte(melhor_solucao, α);
4 |   nova_solucao ← ReconstroiSolucao(solucao_parcial, I);
5 |   if Custo(nova_solucao) < Custo(melhor_solucao) then
6 |   |   melhor_solucao ← nova_solucao;
7 |   end
8 end
9 return melhor_solucao;
```

CAPÍTULO

4

Trabalho Desenvolvido

Este capítulo descreve a metodologia desenvolvida para a resolução do SSP utilizando uma matheurística primal composta por duas etapas: uma fase construtiva, responsável pela geração de soluções iniciais viáveis, e uma fase de otimização, fundamentada no método *Large Neighbourhood Search* (LNS) proposto por Shaw (1998). Essa combinação tem como objetivo explorar de forma eficiente o espaço de soluções, balanceando qualidade e tempo computacional.

Destaca-se, ainda, que parte das melhorias obtidas durante o refinamento da abordagem foi fruto de discussões técnicas realizadas com a **Professora Dra. Edna Ayako Hoshino**, cuja experiência contribuiu significativamente para a consolidação do método aqui apresentado.

4.1 Construção da Solução Inicial Viável

A fase construtiva tem como objetivo gerar uma solução inicial viável que será posteriormente aprimorada pela fase de otimização. Essa etapa foi desenvolvida utilizando a estratégia de construção de soluções do método GRASP (*Greedy Randomized Adaptive Search Procedure*). De acordo com essa abordagem, parte-se de um *safe set* trivial, composto por todos os vértices do grafo, e iterativamente tenta-se remover vértices, mantendo sempre a viabilidade da solução.

O algoritmo adotado tem por base a heurística apresentada por Macambira et al. (2019). Em cada iteração, os vértices candidatos à remoção são ordenados e uma *Restricted Candidate List* (RCL) é formada. Um vértice é então selecionado aleatoriamente dentre aqueles presentes na RCL e provisoriamente

removido do conjunto. A remoção é confirmada apenas se a solução resultante continuar viável. O procedimento é repetido diversas vezes, variando-se a ordem dos vértices considerados, e ao final a melhor solução encontrada é utilizada como ponto de partida para a fase de otimização com o LNS. O Algoritmo 3 apresenta formalmente a etapa de construção da solução inicial adotado neste trabalho, seguindo a estratégia do método GRASP.

Algoritmo 3: Criação da Solução Inicial

Input: Grafo G, conjunto de vértices V, tamanho da RCL k, número de iterações n_{iter}

Output: Melhor solução viável S^*

```
1  S* ← ∅;
2  for iter ← 1 to n_iter do
3      S ← V;
4      C ← V;
5      while C ≠ ∅ do
6          Constrói RCL a partir dos vértices de C;
7          Seleciona aleatoriamente v ← elemento de RCL;
8          S ← S \ {v};
9          if S não é safe set then
10             S ← S ∪ {v};
11         end
12         C ← C \ {v};
13     end
14     if Custo(S) < Custo(S*) then
15         S* ← S;
16     end
17 end
18 return S*;
```

4.2 Fase de Otimização com o LNS

A segunda etapa busca melhorar a solução inicial utilizando a técnica LNS, que é um processo de otimização que alterna entre destruir parcialmente uma solução e reconstruí-la de forma controlada. Este método envolve dois estágios principais:

- **Destruição**: elimina-se uma parte dos vértices da solução atual, gerando uma configuração parcial ou inviável;
- **Reparação**: reconstrói-se a solução ao resolver o problema novamente

para os vértices excluídos, enquanto se mantém inalterados aqueles que não foram afetados.

A resolução do modelo de PLI apresentado por Hosteins (2020) é utilizada neste trabalho como meio de conduzir o processo de reconstrução, fixando as variáveis associadas aos vértices que permaneçam no *safe set* inicial em 1 e aquelas que não pertencem a ele em 0. Dessa forma, apenas as variáveis correspondentes aos vértices destruídos permanecem livres, reduzindo significativamente o espaço de busca durante a etapa de reparação. Essa abordagem diminui o tempo de processamento e o espaço de busca, sem comprometer a consistência e a qualidade das soluções.

No presente trabalho, a implementação do LNS aconteceu da seguinte maneira:

1. O GRASP é empregado na elaboração da solução inicial.

2. Uma parte dessa solução é removida (destruição).

3. A formulação de Hosteins (2020) em PLI (reconstrução) soluciona o problema simplificado.

4. O procedimento se repete, resultando em soluções progressivamente mais robustas.

Essa abordagem possibilita a exploração de vizinhanças muito mais amplas do que as convencionais em buscas locais, ampliando as oportunidades de evitar ótimos locais e descobrir soluções de alta qualidade em um período menor.

4.2.1 Estratégias de Seleção e Destruição de Vértices

Na etapa de destruição do LNS, foram utilizadas duas abordagens diferentes para determinar quais vértices seriam removidos:

- **Vértices da borda**: nesta abordagem, dá-se preferência à escolha de vértices que se encontram nas bordas das componentes do grafo. Um vértice está na borda se é adjacente a um vértice pertencente a outra componente, isto é, se existe uma aresta conectando um vértice do conjunto seguro a um que está fora dele. A Figura 4.1 ilustra um grafo com a indicação dos vértices que estão na borda das componentes e por isso serão tidos como candidatos.

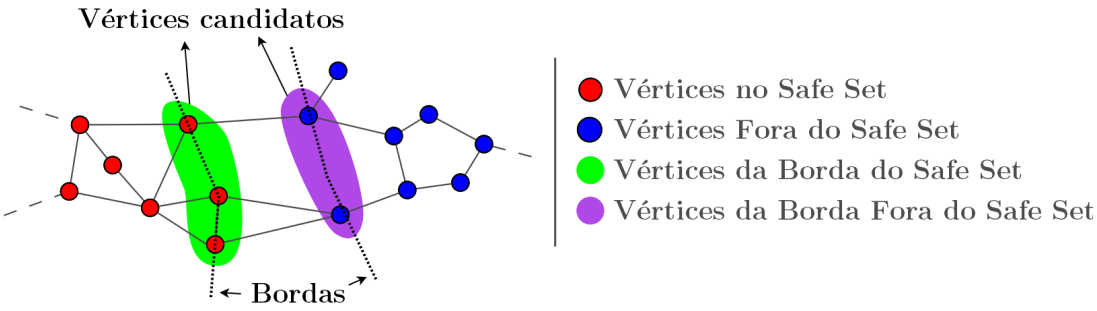

Figura 4.1: Ilustração de estratégia utilizando os vértices da borda como candidatos.

- **Vértices fora da borda**: neste enfoque, são escolhidos vértices que não estão na fronteira das componentes, ou seja, não há uma aresta ligando-os a componentes a eles adjacentes. A Figura 4.2 ilustra um grafo com a indicação dos vértices que serão os candidatos por estarem fora da borda das componentes.

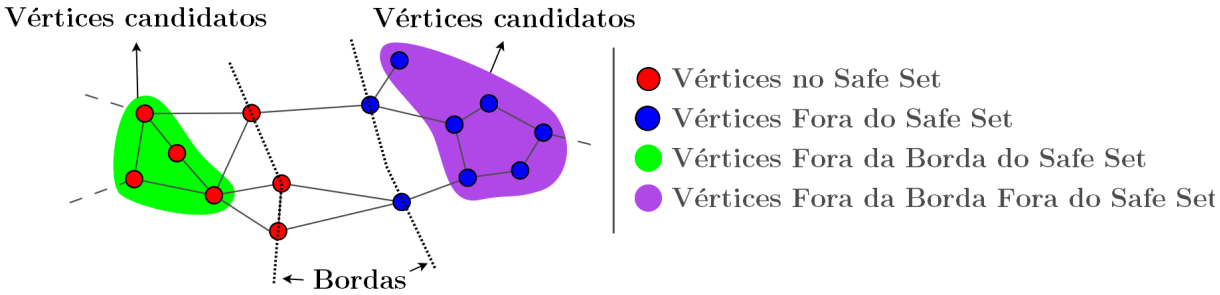

Figura 4.2: Ilustração de estratégia utilizando os vértices fora da borda como candidatos.

A seleção dos vértices potenciais para a destruição é guiada pela Lista Restrita de Candidatos (RCL), que consiste nos vértices j que satisfazem a seguinte condição:

$$w(j) \geq \alpha w_{\max} + (1-\alpha) w_{\min} \tag{4.1}$$

sendo $w_{\max}$ e $w_{\min}$ os pesos máximos e mínimos entre os vértices candidatos, respectivamente, e $\alpha \in [0,1]$ é o parâmetro que controla o grau de aleatoriedade. Em nossos experimentos, adotamos $\alpha = 0,7$ Depois que a RCL é criada, um vértice é selecionado aleatoriamente e retirado da solução. Esse procedimento é continuado até que seja removida metade dos vértices candidatos.

CAPÍTULO

5

Resultados Computacionais

Este capítulo apresenta o ambiente e os resultados computacionais obtidos através do modelo desenvolvido. Com o objetivo de avaliar o desempenho da matheurística proposta, foram realizados experimentos em instâncias da literatura, de modo a analisar a qualidade das soluções encontradas e o comportamento do algoritmo sob diferentes configurações de parâmetros.

5.1 Ambiente computacional

Todos os experimentos foram executados em um ambiente controlado, em máquina com processador Intel(R) Core(TM) i5-10210U (1,60 GHz), 20 GB de memória RAM e sistema operacional Linux. A implementação foi desenvolvida na linguagem C, compilada com GCC versão 11.4.0, utilizando o resolvedor SCIP v8.1.0 em modo *single thread*. Para assegurar comparabilidade entre os testes, o tempo máximo de execução foi fixado em 10 s para a fase construtiva e 25 s para o procedimento LNS.

5.2 Conjunto de Instâncias

Foram realizados testes computacionais com 120 grafos pseudoaleatórios conectados gerados por Hosteins (2020), que possuem entre 20 e 50 vértices e densidades que variam de 10% a 40% em relação ao número de arestas de um grafo completo. Esses casos foram selecionados por apresentarem distintos níveis de complexidade estrutural, o que possibilita analisar como o tamanho e a densidade do grafo afetam o desempenho da matheurística.

5.3 Configurações experimentais

Quatro diferentes configurações do algoritmo foram testadas, com variações no número de iterações da fase construtiva e na forma de escolher os vértices na fase de destruição:

- **B-1 e B-10**: utiliza os vértices da borda como candidatos à remoção. Na versão B-1, a melhor solução inicial é escolhida em apenas uma execução da fase construtiva. Na versão B-10, é escolhida dentre dez iterações independentes.

- **F-1 e F-10**: apenas vértices fora da borda são usados como candidatos à destruição. Na fase construtiva, a versão F-1 do algoritmo escolhe a melhor solução inicial com apenas uma execução da fase construtiva e, na versão F-10, o algoritmo escolhe dentre dez iterações.

Nos experimentos realizados foram considerados também uma configuração inicial, denotada por **I-1**, que consiste na criação de uma solução inicial, com o emprego de uma iteração do GRASP, sem aplicação posterior do algoritmo LNS.

Os resultados obtidos representam a razão entre o acréscimo da solução encontrada em relação à melhor solução obtida por Hosteins (2020). Dessa forma, valores menores indicam soluções mais próximas do ótimo.

5.4 Resultados obtidos por densidade de instâncias

A Tabela 5.1 apresenta a média dos resultados obtidos agrupados pela densidade das instâncias (d), considerando cada configuração experimental.

Tabela 5.1: Comparação com o desempenho do algoritmo exato por densidade.

d	I-1	B-1	F-1	B-10	F-10
10%	137,2%	113,2%	108,3%	86,0%	84,4%
20%	82,5%	60,7%	58,6%	42,6%	44,0%
30%	53,2%	32,9%	33,5%	30,2%	25,1%
40%	36,5%	17,5%	18,5%	12,7%	12,6%

Observa-se que o desempenho melhora substancialmente à medida que a densidade do grafo aumenta. As configurações B-10 e F-10, que realizam quantidades maiores de iterações construtivas, produziram resultados melhores, com reduções médias de até 50% no valor da solução em relação à inicial.

5.5 Resultados obtidos por tamanho das instâncias

A Tabela 5.2 apresenta os resultados médios agrupados pelo número de vértices (n) das instâncias.

Tabela 5.2: Comparação com o desempenho do algoritmo exato por tamanho.

n	I-1	B-1	F-1	B-10	F-10
20	87,1%	57,4%	66,5%	47,1%	46,3%
25	76,4%	61,6%	55,7%	47,3%	45,4%
30	86,5%	69,2%	66,1%	51,4%	51,4%
35	91,1%	64,5%	60,1%	44,6%	41,0%
40	68,3%	49,7%	48,0%	38,1%	36,3%
50	54,6%	34,0%	32,0%	28,9%	28,8%

De modo análogo ao comportamento observado pela densidade, as melhorias foram mais expressivas em grafos de maior dimensão, nos quais o espaço de busca é mais amplo. As configurações com múltiplas iterações (B-10 e F-10) apresentaram desempenho superior em todos os tamanhos, demonstrando que o refinamento iterativo da solução inicial potencializa o efeito do LNS. Além disso, o uso de vértices fora da borda (F-10) manteve-se levemente melhor em média, alcançando as melhores soluções em aproximadamente 77% das instâncias testadas.

5.6 Análise dos resultados

Os resultados demonstram que a matheurística proposta apresenta desempenho consistente, especialmente em instâncias de maior complexidade. A etapa de otimização por LNS proporcionou melhoria média de 50% no valor da solução inicial, evidenciando sua eficiência em explorar grandes vizinhanças e escapar de ótimos locais.

Observou-se também que as estratégias avaliadas apresentaram melhor desempenho em instâncias maiores e mais densas, fornecendo soluções de boa qualidade em frações do tempo computacional demandado por métodos exatos.

Além disso, em termos qualitativos, verificou-se que o número de iterações na fase construtiva exerce influência direta sobre a qualidade final das soluções: um maior número de iterações tende a gerar soluções iniciais melhores, o que favorece o trabalho do método LNS. Nesse sentido, as configurações com dez iterações na fase construtiva (F-10 e B-10) destacaram-se por produzir o melhor desempenho entre as configurações testadas.

Por fim, destaca-se que a aplicação do LNS contribuiu expressivamente para a melhoria das soluções iniciais, confirmando sua eficácia como método

de melhoria das soluções em combinação com a fase construtiva, tendo a estratégia de seleção de candidatos fora da borda se mostrado ligeiramente superior, por promover maior diversidade nas soluções geradas e, consequentemente, melhor desempenho médio em relação às demais configurações.

CAPÍTULO

6

Conclusões e Considerações Finais

Este trabalho apresentou o desenvolvimento e a avaliação de uma matheurística aplicada ao Problema do *Safe Set*, combinando uma fase construtiva baseada em heurísticas gulosas aleatorizadas com um processo de refinamento conduzido pelo método *Large Neighbourhood Search*. O objetivo foi obter soluções viáveis e de boa qualidade em tempo computacional reduzido, considerando a natureza NP-difícil do problema.

Os resultados obtidos mostraram que a abordagem proposta é capaz de melhorar significativamente as soluções iniciais, reduzindo o valor da solução em cerca de 50%. As configurações com dez iterações na fase construtiva (F-10 e B-10) apresentaram o melhor desempenho, e a estratégia de destruição fora da borda mostrou-se ligeiramente superior por proporcionar maior diversidade na busca.

De modo geral, a matheurística proposta demonstrou ser uma alternativa interessante e de bom custo-benefício para o SSP, fornecendo soluções próximas do ótimo em frações do tempo exigido por métodos exatos.

Como continuidade deste trabalho, recomenda-se investigar novos critérios de destruição e reconstrução no LNS e explorar ajustes automáticos de parâmetros para aprimorar ainda mais o desempenho do método.

Referências Bibliográficas

Bapat, R., Fujita, S., Legay, S., Manoussakis, Y., Matsui, Y., Sakuma, T., e Tuza, Z. (2016). Network majority on tree topological network. *Electron. Notes Discrete Math.*, 54:79–84. Citado nas páginas 2 e 3.

Feo, T. A. e Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2): 67–71. Citado na página 11.

Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133. Citado na página 11.

Fujita, S., MacGillivray, G., e Sakuma, T. (2016). Safe set problem on graphs. *Discrete Applied Mathematics*, 215:106–111. Citado nas páginas 2 e 3.

Hosteins, P. (2020). A compact mixed integer linear formulation for safe set problems. *Optimization Letters*, 14:2127–2148. Citado nas páginas 3, 5, 15, 17, e 18.

Macambira, A. F. U., Simonetti, L., Barbalho, H., Gonzalez, P. H., e Maculan, N. (2019). A new formulation for the safe set problem on graphs. *Computers and Operations Research*, 111:346–356. Citado nas páginas 3 e 13.

Malaguti, E. e Pedrotti, V. (2023). Models and algorithms for the weighted safe set problem. *Discrete Applied Mathematics*, 329:23–34. ISSN 0166-218X. URL https://www.sciencedirect.com/science/article/pii/S0166218X22004863. Citado na página 3.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. e Puget, J.-F., editors, *Principles and Practice of Constraint Programming — CP98*, p. 417–431, Berlin, Heidelberg. Springer. ISBN 978-3-540-49481-2. Citado nas páginas 12 e 13.

APÊNDICE

A

Resultados computacionais detalhados

Neste anexo são apresentados os resultados computacionais detalhados por instância. A Tabela A.1 contém, para cada instância da base de testes, o valor ótimo (obtido pelo algoritmo exato) e as soluções produzidas pelo algoritmo proposto nas diferentes configurações consideradas (I-1, B-1, F-1, B-10 e F-10). A Tabela A.2 apresenta, para as mesmas instâncias e configurações, o *gap* percentual entre o valor ótimo e as soluções encontradas, permitindo avaliar de forma quantitativa a qualidade relativa das soluções produzidas por cada configuração.

Os nomes das instâncias seguem o padrão adotado na base de testes e refletem suas principais características, incluindo o tamanho do grafo, a densidade utilizada na geração e o índice da instância dentro de cada grupo. As densidades são representadas por identificadores do tipo r, em que r1 corresponde a 10% de densidade, r2 a 20%, r3 a 30% e r4 a 40%, respectivamente.

Foram avaliadas cinco configurações distintas. A configuração I-1 representa o valor da solução inicial obtida após uma única iteração do procedimento de construção, sem aplicação posterior do LNS. Nas configurações B-1 e F-1 a solução inicial viável é também produzida por uma única iteração do respectivo procedimento construtivo, enquanto nas configurações B-10 e F-10 seleciona-se a melhor solução dentre 10 iterações independentes. Para a etapa de LNS, as configurações B-1 e B-10 utilizam como candidatos os vértices pertencentes à borda, ao passo que F-1 e F-10 consideram os vértices fora da borda.

Tabela A.1: Valores ótimos e soluções obtidas nas diferentes configurações.

Instâncias	**Solução Ótima**	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
20_r1_1.ninst	43	82	60	78	69	69
20_r1_2.ninst	40	84	81	71	70	70
20_r1_3.ninst	46	84	66	84	65	76
20_r1_4.ninst	43	99	74	88	70	76
20_r1_5.ninst	41	81	76	72	75	75
20_r2_1.ninst	26	74	69	59	35	36
20_r2_2.ninst	33	66	49	61	60	57
20_r2_3.ninst	23	50	48	47	35	39
20_r2_4.ninst	26	50	45	45	31	41
20_r2_5.ninst	31	58	48	53	45	45
20_r3_1.ninst	33	62	56	58	49	52
20_r3_2.ninst	39	67	49	58	54	51
20_r3_3.ninst	36	68	63	54	45	42
20_r3_4.ninst	31	48	46	48	47	47
20_r3_5.ninst	40	73	54	60	94	48
20_r4_1.ninst	43	64	58	58	53	52
20_r4_2.ninst	49	76	57	66	56	58
20_r4_3.ninst	42	68	45	58	50	51
20_r4_4.ninst	50	76	59	65	59	59
20_r4_5.ninst	48	70	60	63	54	61
25_r1_1.ninst	54	99	85	85	85	85

Continua na próxima página

Tabela A.1 — Continuação da página anterior.

Instâncias	**Solução Ótima**	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
25_r1_2.ninst	43	92	86	79	86	86
25_r1_3.ninst	43	76	76	76	74	71
25_r1_4.ninst	33	96	86	86	80	80
25_r1_5.ninst	48	120	118	110	83	83
25_r2_1.ninst	39	81	69	66	64	64
25_r2_2.ninst	44	92	76	68	70	70
25_r2_3.ninst	35	68	58	58	48	48
25_r2_4.ninst	53	90	87	79	75	75
25_r2_5.ninst	38	65	65	65	53	53
25_r3_1.ninst	48	86	71	77	66	66
25_r3_2.ninst	57	91	83	74	72	73
25_r3_3.ninst	62	83	79	79	77	77
25_r3_4.ninst	53	74	74	71	77	69
25_r3_5.ninst	48	84	69	69	64	64
25_r4_1.ninst	57	68	87	68	72	66
25_r4_2.ninst	71	92	82	80	80	80
25_r4_3.ninst	56	72	68	68	67	67
25_r4_4.ninst	57	72	64	64	68	64
25_r4_5.ninst	54	92	73	73	62	62
30_r1_1.ninst	51	136	133	129	97	97
30_r1_2.ninst	43	109	104	100	89	89
30_r1_3.ninst	40	112	104	95	100	100

Continua na próxima página

Tabela A.1 — Continuação da página anterior.

Instâncias	**Solução Ótima**	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
30_r1_4.ninst	49	129	116	116	102	102
30_r1_5.ninst	38	89	89	89	72	72
30_r2_1.ninst	54	110	103	103	79	79
30_r2_2.ninst	52	107	91	91	82	82
30_r2_3.ninst	50	94	79	71	74	74
30_r2_4.ninst	51	101	88	84	72	72
30_r2_5.ninst	50	88	82	82	78	78
30_r3_1.ninst	64	97	83	83	77	77
30_r3_2.ninst	59	87	80	80	79	79
30_r3_3.ninst	70	113	90	91	91	91
30_r3_4.ninst	45	78	71	71	63	63
30_r3_5.ninst	68	97	89	89	89	89
30_r4_1.ninst	71	102	89	89	85	85
30_r4_2.ninst	72	106	92	92	81	81
30_r4_3.ninst	76	100	85	85	85	85
30_r4_4.ninst	68	96	86	86	82	82
30_r4_5.ninst	73	88	85	85	83	83
35_r1_1.ninst	45	144	134	119	95	91
35_r1_2.ninst	54	136	120	120	95	95
35_r1_3.ninst	45	112	105	92	93	85
35_r1_4.ninst	39	119	111	111	89	83
35_r1_5.ninst	46	133	113	104	77	67

Continua na próxima página

Tabela A.1 — Continuação da página anterior.

Instâncias	**Solução Ótima**	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
35_r2_1.ninst	61	117	97	93	87	87
35_r2_2.ninst	73	127	107	107	104	104
35_r2_3.ninst	64	114	97	97	94	91
35_r2_4.ninst	51	93	86	86	81	81
35_r2_5.ninst	73	131	112	112	100	99
35_r3_1.ninst	86	127	103	103	103	103
35_r3_2.ninst	88	127	114	114	109	107
35_r3_3.ninst	83	123	104	104	100	100
35_r3_4.ninst	78	138	104	104	96	96
35_r3_5.ninst	66	122	95	95	94	94
35_r4_1.ninst	93	129	106	106	99	99
35_r4_2.ninst	93	131	108	108	100	100
35_r4_3.ninst	96	136	112	112	105	105
35_r4_4.ninst	85	113	95	95	94	94
35_r4_5.ninst	95	138	111	111	105	105
40_r1_1.ninst	64	120	112	112	112	112
40_r1_2.ninst	54	134	117	117	95	95
40_r1_3.ninst	61	156	137	127	112	105
40_r1_4.ninst	52	126	116	116	89	87
40_r1_5.ninst	53	121	116	108	99	99
40_r2_1.ninst	82	145	117	117	116	116
40_r2_2.ninst	76	141	126	126	115	112

Continua na próxima página

Tabela A.1 — Continuação da página anterior.

Instâncias	**Solução Ótima**	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
40_r2_3.ninst	68	107	98	97	96	96
40_r2_4.ninst	62	115	104	104	96	96
40_r2_5.ninst	75	120	107	107	101	101
40_r3_1.ninst	95	129	119	119	110	110
40_r3_2.ninst	108	137	126	126	124	124
40_r3_3.ninst	94	112	109	109	108	108
40_r3_4.ninst	84	133	110	110	120	108
40_r3_5.ninst	105	149	128	128	120	118
40_r4_1.ninst	98	134	110	110	105	105
40_r4_2.ninst	96	120	105	105	105	105
40_r4_3.ninst	105	135	118	118	114	114
40_r4_4.ninst	108	150	120	120	118	118
40_r4_5.ninst	93	118	108	108	101	101
50_r1_1.ninst	75	185	157	149	142	142
50_r1_2.ninst	82	183	151	144	136	136
50_r1_3.ninst	61	144	120	119	119	119
50_r1_4.ninst	92	176	153	153	145	145
50_r1_5.ninst	82	178	149	133	146	146
50_r2_1.ninst	110	164	146	146	142	142
50_r2_2.ninst	129	192	155	155	151	151
50_r2_3.ninst	106	138	130	130	130	130
50_r2_4.ninst	117	170	146	146	140	140

Continua na próxima página

Tabela A.1 — Continuação da página anterior.

Instâncias	**Solução Ótima**	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
50_r2_5.ninst	136	173	160	160	157	157
50_r3_1.ninst	119	153	135	135	133	133
50_r3_2.ninst	120	164	138	138	132	132
50_r3_3.ninst	122	169	142	142	132	132
50_r3_4.ninst	114	154	142	142	130	130
50_r3_5.ninst	113	138	128	128	125	125
50_r4_1.ninst	132	158	141	141	137	137
50_r4_2.ninst	144	182	156	156	152	152
50_r4_3.ninst	125	168	137	137	135	135
50_r4_4.ninst	126	153	136	136	138	135
50_r4_5.ninst	117	134	127	127	126	126

Fim da tabela

Tabela A.2: Gap percentual entre o valor ótimo, determinado pelo algoritmo exato, e o valor da solução produzida pelo algoritmo proposto, em suas diversas configurações, para cada instância da base de testes.

Instâncias	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
20_r1_1.ninst	90,70%	39,53%	81,40%	60,47%	60,47%
20_r1_2.ninst	110,00%	102,50%	77,50%	75,00%	75,00%
20_r1_3.ninst	82,61%	43,48%	82,61%	41,30%	65,22%
20_r1_4.ninst	130,23%	72,09%	104,65%	62,79%	76,74%
20_r1_5.ninst	97,56%	85,37%	75,61%	82,93%	82,93%

Continua na próxima página

Tabela A.2 — Continuação da página anterior.

Instâncias	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
20_r2_1.ninst	184,62%	165,38%	126,92%	34,62%	38,46%
20_r2_2.ninst	100,00%	48,48%	84,85%	81,82%	72,73%
20_r2_3.ninst	117,39%	108,70%	104,35%	52,17%	69,57%
20_r2_4.ninst	92,31%	73,08%	73,08%	19,23%	57,69%
20_r2_5.ninst	87,10%	54,84%	70,97%	45,16%	45,16%
20_r3_1.ninst	87,88%	69,70%	75,76%	48,48%	57,58%
20_r3_2.ninst	71,79%	25,64%	48,72%	38,46%	30,77%
20_r3_3.ninst	88,89%	75,00%	50,00%	25,00%	16,67%
20_r3_4.ninst	54,84%	48,39%	54,84%	51,61%	51,61%
20_r3_5.ninst	82,50%	35,00%	50,00%	135,00%	20,00%
20_r4_1.ninst	48,84%	34,88%	34,88%	23,26%	20,93%
20_r4_2.ninst	55,10%	16,33%	34,69%	14,29%	18,37%
20_r4_3.ninst	61,90%	7,14%	38,10%	19,05%	21,43%
20_r4_4.ninst	52,00%	18,00%	30,00%	18,00%	18,00%
20_r4_5.ninst	45,83%	25,00%	31,25%	12,50%	27,08%
25_r1_1.ninst	83,33%	57,41%	57,41%	57,41%	57,41%
25_r1_2.ninst	113,95%	100,00%	83,72%	100,00%	100,00%
25_r1_3.ninst	76,74%	76,74%	76,74%	72,09%	65,12%
25_r1_4.ninst	190,91%	160,61%	160,61%	142,42%	142,42%
25_r1_5.ninst	150,00%	145,83%	129,17%	72,92%	72,92%
25_r2_1.ninst	107,69%	76,92%	69,23%	64,10%	64,10%
25_r2_2.ninst	109,09%	72,73%	54,55%	59,09%	59,09%
25_r2_3.ninst	94,29%	65,71%	65,71%	37,14%	37,14%

Continua na próxima página

Tabela A.2 — Continuação da página anterior.

Instâncias	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
25_r2_4.ninst	69,81%	64,15%	49,06%	41,51%	41,51%
25_r2_5.ninst	71,05%	71,05%	71,05%	39,47%	39,47%
25_r3_1.ninst	79,17%	47,92%	60,42%	37,50%	37,50%
25_r3_2.ninst	59,65%	45,61%	29,82%	26,32%	28,07%
25_r3_3.ninst	33,87%	27,42%	27,42%	24,19%	24,19%
25_r3_4.ninst	39,62%	39,62%	33,96%	45,28%	30,19%
25_r3_5.ninst	75,00%	43,75%	43,75%	33,33%	33,33%
25_r4_1.ninst	19,30%	52,63%	19,30%	26,32%	15,79%
25_r4_2.ninst	29,58%	15,49%	12,68%	12,68%	12,68%
25_r4_3.ninst	28,57%	21,43%	21,43%	19,64%	19,64%
25_r4_4.ninst	26,32%	12,28%	12,28%	19,30%	12,28%
25_r4_5.ninst	70,37%	35,19%	35,19%	14,81%	14,81%
30_r1_1.ninst	166,67%	160,78%	152,94%	90,20%	90,20%
30_r1_2.ninst	153,49%	141,86%	132,56%	106,98%	106,98%
30_r1_3.ninst	180,00%	160,00%	137,50%	150,00%	150,00%
30_r1_4.ninst	163,27%	136,73%	136,73%	108,16%	108,16%
30_r1_5.ninst	134,21%	134,21%	134,21%	89,47%	89,47%
30_r2_1.ninst	103,70%	90,74%	90,74%	46,30%	46,30%
30_r2_2.ninst	105,77%	75,00%	75,00%	57,69%	57,69%
30_r2_3.ninst	88,00%	58,00%	42,00%	48,00%	48,00%
30_r2_4.ninst	98,04%	72,55%	64,71%	41,18%	41,18%
30_r2_5.ninst	76,00%	64,00%	64,00%	56,00%	56,00%
30_r3_1.ninst	51,56%	29,69%	29,69%	20,31%	20,31%

Continua na próxima página

Tabela A.2 — Continuação da página anterior.

Instâncias	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
30_r3_2.ninst	47,46%	35,59%	35,59%	33,90%	33,90%
30_r3_3.ninst	61,43%	28,57%	30,00%	30,00%	30,00%
30_r3_4.ninst	73,33%	57,78%	57,78%	40,00%	40,00%
30_r3_5.ninst	42,65%	30,88%	30,88%	30,88%	30,88%
30_r4_1.ninst	43,66%	25,35%	25,35%	19,72%	19,72%
30_r4_2.ninst	47,22%	27,78%	27,78%	12,50%	12,50%
30_r4_3.ninst	31,58%	11,84%	11,84%	11,84%	11,84%
30_r4_4.ninst	41,18%	26,47%	26,47%	20,59%	20,59%
30_r4_5.ninst	20,55%	16,44%	16,44%	13,70%	13,70%
35_r1_1.ninst	220,00%	197,78%	164,44%	111,11%	102,22%
35_r1_2.ninst	151,85%	122,22%	122,22%	75,93%	75,93%
35_r1_3.ninst	148,89%	133,33%	104,44%	106,67%	88,89%
35_r1_4.ninst	205,13%	184,62%	184,62%	128,21%	112,82%
35_r1_5.ninst	189,13%	145,65%	126,09%	67,39%	45,65%
35_r2_1.ninst	91,80%	59,02%	52,46%	42,62%	42,62%
35_r2_2.ninst	73,97%	46,58%	46,58%	42,47%	42,47%
35_r2_3.ninst	78,13%	51,56%	51,56%	46,88%	42,19%
35_r2_4.ninst	82,35%	68,63%	68,63%	58,82%	58,82%
35_r2_5.ninst	79,45%	53,42%	53,42%	36,99%	35,62%
35_r3_1.ninst	47,67%	19,77%	19,77%	19,77%	19,77%
35_r3_2.ninst	44,32%	29,55%	29,55%	23,86%	21,59%
35_r3_3.ninst	48,19%	25,30%	25,30%	20,48%	20,48%
35_r3_4.ninst	76,92%	33,33%	33,33%	23,08%	23,08%

Continua na próxima página

Tabela A.2 — Continuação da página anterior.

Instâncias	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
35_r3_5.ninst	84,85%	43,94%	43,94%	42,42%	42,42%
35_r4_1.ninst	38,71%	13,98%	13,98%	6,45%	6,45%
35_r4_2.ninst	40,86%	16,13%	16,13%	7,53%	7,53%
35_r4_3.ninst	41,67%	16,67%	16,67%	9,38%	9,38%
35_r4_4.ninst	32,94%	11,76%	11,76%	10,59%	10,59%
35_r4_5.ninst	45,26%	16,84%	16,84%	10,53%	10,53%
40_r1_1.ninst	87,50%	75,00%	75,00%	75,00%	75,00%
40_r1_2.ninst	148,15%	116,67%	116,67%	75,93%	75,93%
40_r1_3.ninst	155,74%	124,59%	108,20%	83,61%	72,13%
40_r1_4.ninst	142,31%	123,08%	123,08%	71,15%	67,31%
40_r1_5.ninst	128,30%	118,87%	103,77%	86,79%	86,79%
40_r2_1.ninst	76,83%	42,68%	42,68%	41,46%	41,46%
40_r2_2.ninst	85,53%	65,79%	65,79%	51,32%	47,37%
40_r2_3.ninst	57,35%	44,12%	42,65%	41,18%	41,18%
40_r2_4.ninst	85,48%	67,74%	67,74%	54,84%	54,84%
40_r2_5.ninst	60,00%	42,67%	42,67%	34,67%	34,67%
40_r3_1.ninst	35,79%	25,26%	25,26%	15,79%	15,79%
40_r3_2.ninst	26,85%	16,67%	16,67%	14,81%	14,81%
40_r3_3.ninst	19,15%	15,96%	15,96%	14,89%	14,89%
40_r3_4.ninst	58,33%	30,95%	30,95%	42,86%	28,57%
40_r3_5.ninst	41,90%	21,90%	21,90%	14,29%	12,38%
40_r4_1.ninst	36,73%	12,24%	12,24%	7,14%	7,14%
40_r4_2.ninst	25,00%	9,38%	9,38%	9,38%	9,38%

Continua na próxima página

Tabela A.2 — Continuação da página anterior.

Instâncias	**I-1**	**B-1**	**F-1**	**B-10**	**F-10**
40_r4_3.ninst	28,57%	12,38%	12,38%	8,57%	8,57%
40_r4_4.ninst	38,89%	11,11%	11,11%	9,26%	9,26%
40_r4_5.ninst	26,88%	16,13%	16,13%	8,60%	8,60%
50_r1_1.ninst	146,67%	109,33%	98,67%	89,33%	89,33%
50_r1_2.ninst	123,17%	84,15%	75,61%	65,85%	65,85%
50_r1_3.ninst	136,07%	96,72%	95,08%	95,08%	95,08%
50_r1_4.ninst	91,30%	66,30%	66,30%	57,61%	57,61%
50_r1_5.ninst	117,07%	81,71%	81,71%	78,05%	78,05%
50_r2_1.ninst	49,09%	32,73%	32,73%	29,09%	29,09%
50_r2_2.ninst	48,84%	20,16%	20,16%	17,05%	17,05%
50_r2_3.ninst	30,19%	22,64%	22,64%	22,64%	22,64%
50_r2_4.ninst	45,30%	24,79%	24,79%	19,66%	19,66%
50_r2_5.ninst	27,21%	17,65%	17,65%	15,44%	15,44%
50_r3_1.ninst	28,57%	13,45%	13,45%	11,76%	11,76%
50_r3_2.ninst	36,67%	15,00%	15,00%	10,00%	10,00%
50_r3_3.ninst	38,52%	16,39%	16,39%	8,20%	8,20%
50_r3_4.ninst	35,09%	24,56%	24,56%	14,04%	14,04%
50_r3_5.ninst	22,12%	13,27%	13,27%	10,62%	10,62%
50_r4_1.ninst	19,70%	6,82%	6,82%	3,79%	3,79%
50_r4_2.ninst	26,39%	8,33%	8,33%	5,56%	5,56%
50_r4_3.ninst	34,40%	9,60%	9,60%	8,00%	8,00%
50_r4_4.ninst	21,43%	7,94%	7,94%	9,52%	7,14%
50_r4_5.ninst	14,53%	8,55%	8,55%	7,69%	7,69%
Fim da tabela					